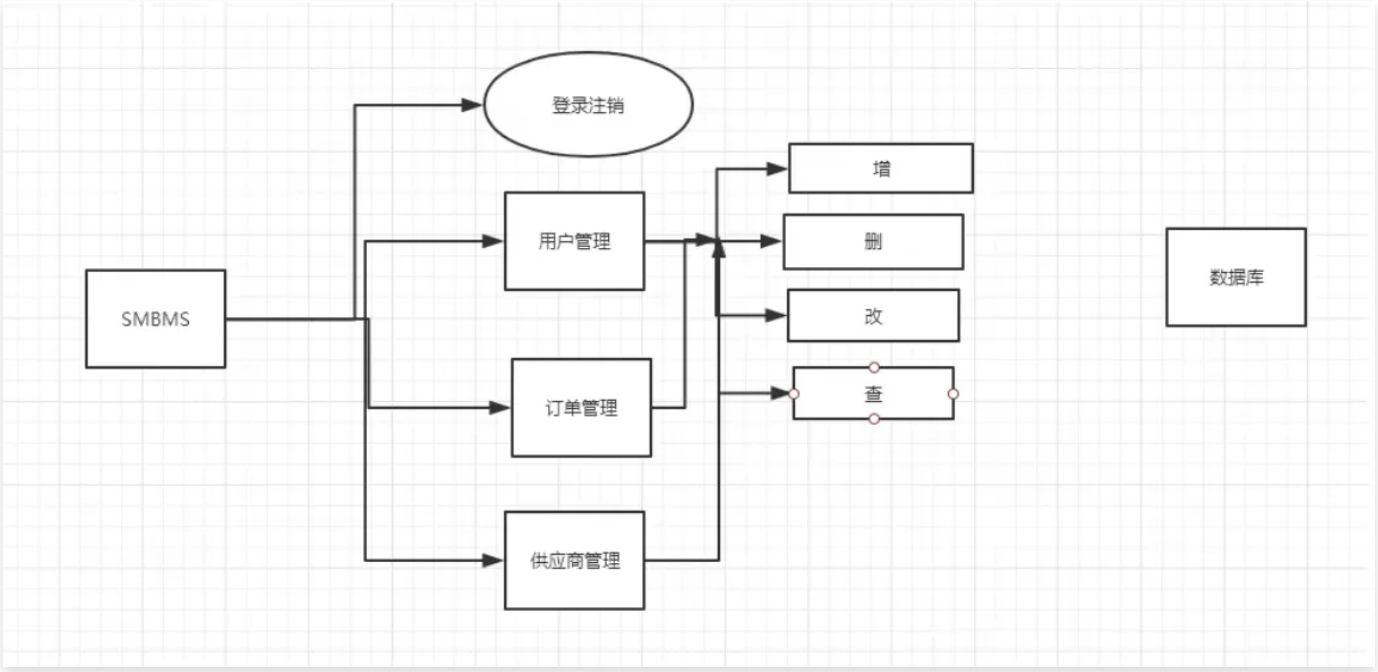
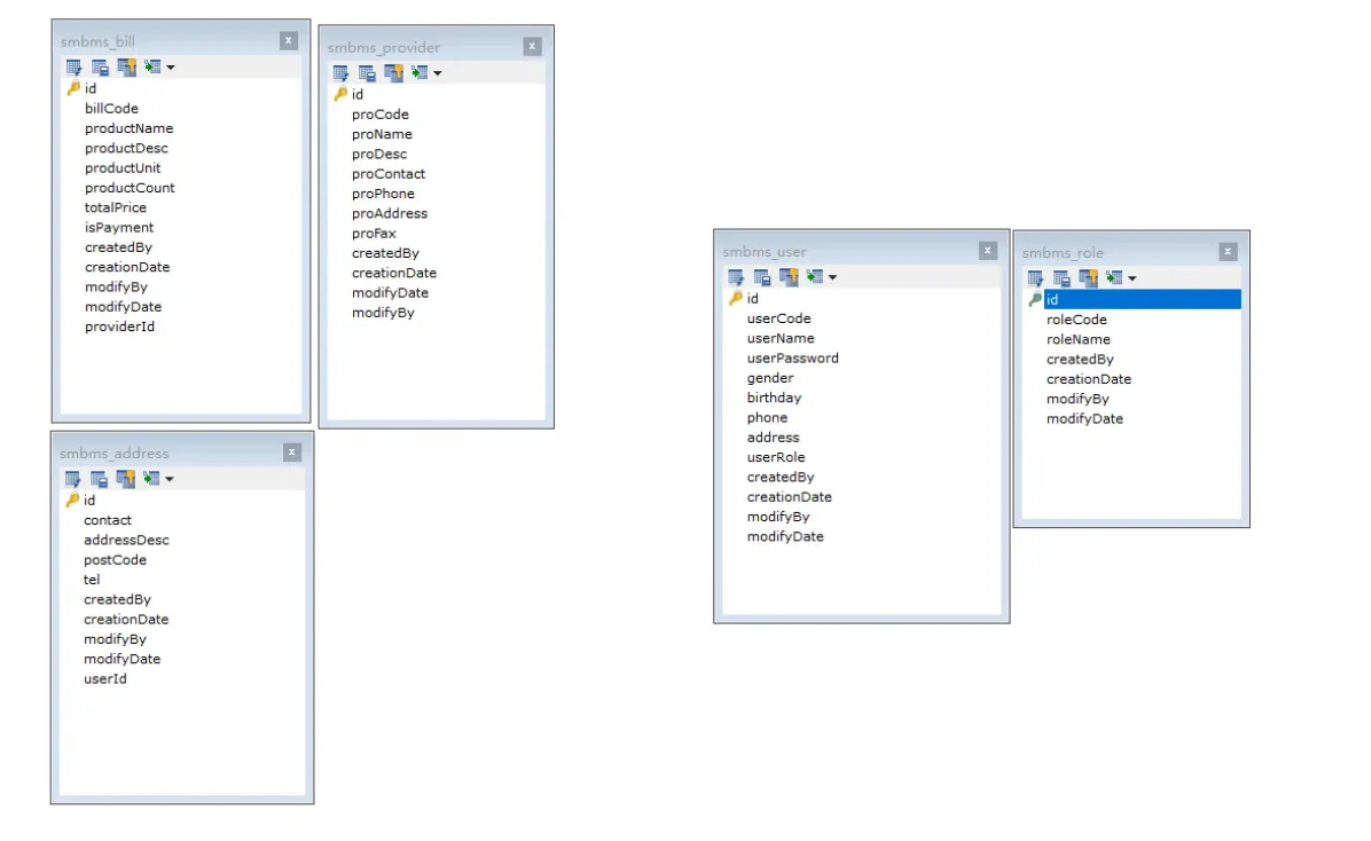
smbms

框架



数据库



项目如何搭建

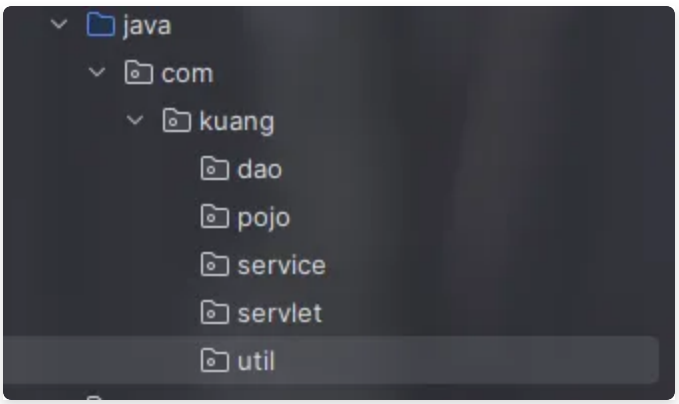
考虑使不使用maven，依赖与jar包

项目搭建准备工作

1. 搭建一个maven web项目
2. 配置tomcat
3. 测试项目是否能够跑起来
4. 导入项目中会遇到的jar包

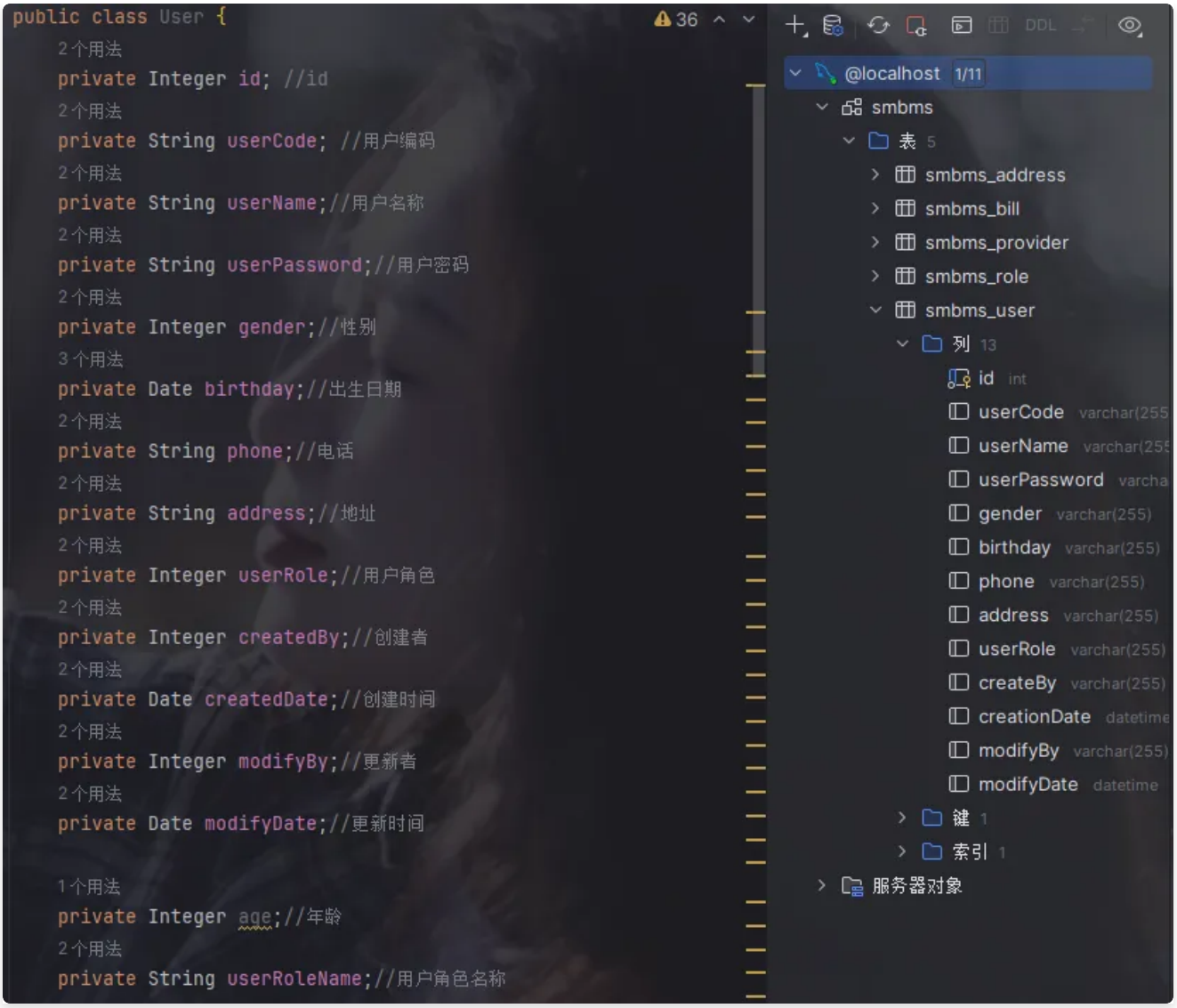
```xml
//jsp、servlet、mysql驱动、jstl、stand....
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
</dependency>

<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>javax.servlet.jsp-api</artifactId>
  <version>2.3.3</version>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.25</version>
</dependency>
```

5. 创建项目包结构

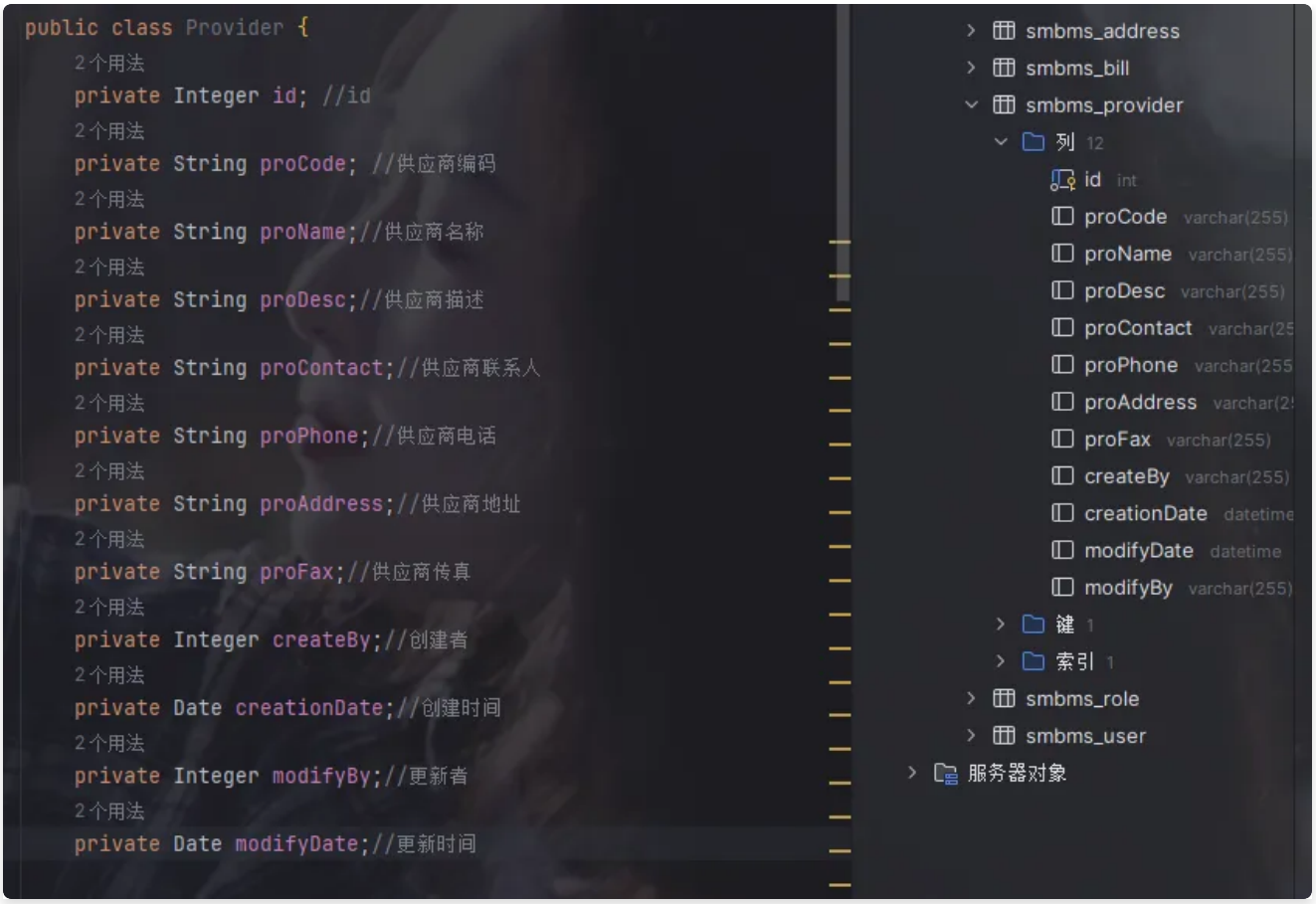


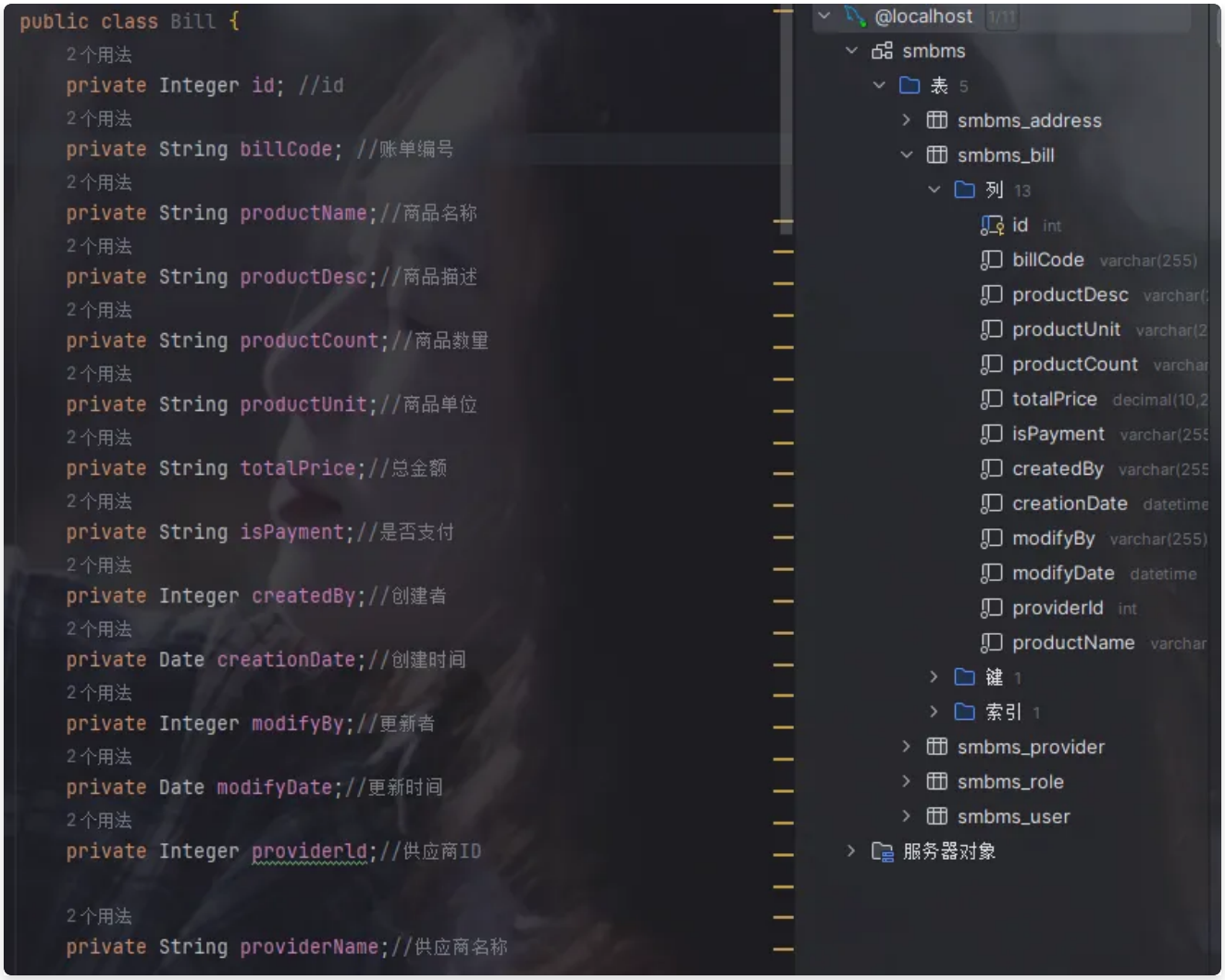
6. 编写实体类

ORM映射：表-类映射


public class User {
36
2个用法
private Integer id; //id
2个用法
private String userCode; //用户编码
2个用法
private String userName;//用户名称
2个用法
private String userPassword;//用户密码
2个用法
private Integer gender;//性别
3个用法
private Date birthday;//出生日期
2个用法
private String phone;//电话
2个用法
private String address;//地址
2个用法
private Integer userRole;//用户角色
2个用法
private Integer createdBy;//创建者
2个用法
private Date createdDate;//创建时间
2个用法
private Integer modifyBy;//更新者
2个用法
private Date modifyDate;//更新时间
1个用法
private Integer age;//年龄
2个用法
private String userRoleName;//用户角色名称
DDL
@localhost 1/11
smbms
表 5
smbms_address
smbms_bill
smbms_provider
smbms_role
smbms_user
列 13
id int
userCode varchar(255
userName varchar(255
userPassword varcha
gender varchar(255)
birthday varchar(255)
phone varchar(255)
address varchar(255)
userRole varchar(255)
createBy varchar(255)
creationDate datetime
modifyBy varchar(255)
modifyDate datetime
键 1
索引 1
服务器对象


public class Role {
2个用法
private Integer id;//id
2个用法
private String roleCode;//角色编码
2个用法
private String roleName;//角色名称
2个用法
private Integer createBy;//创建者
2个用法
private String creationDate;//创建时间
2个用法
private Integer modifyBy;//更新者
2个用法
private String modifyDate;//更新时间
smbms
表 5
smbms_address
smbms_bill
smbms_provider
smbms_role
列 7
id int
roleCode varchar(255)
roleName varchar(255
createBy varchar(255)
creationDate datetime
modifyBy varchar(255)
modifyDate datetime
键 1


public class Provider {
2个用法
private Integer id; //id
2个用法
private String proCode; //供应商编码
2个用法
private String proName;//供应商名称
2个用法
private String proDesc;//供应商描述
2个用法
private String proContact;//供应商联系人
2个用法
private String proPhone;//供应商电话
2个用法
private String proAddress;//供应商地址
2个用法
private String proFax;//供应商传真
2个用法
private Integer createBy;//创建者
2个用法
private Date creationDate;//创建时间
2个用法
private Integer modifyBy;//更新者
2个用法
private Date modifyDate;//更新时间
smbms_address
smbms_bill
smbms_provider
列 12
id int
proCode varchar(255)
proName varchar(255)
proDesc varchar(255)
proContact varchar(25
proPhone varchar(255
proAddress varchar(2
proFax varchar(255)
createBy varchar(255)
creationDate datetime
modifyDate datetime
modifyBy varchar(255)
键 1
索引 1
smbms_role
smbms_user
服务器对象



然后再生成对应的get和set方法

7. 编写基础公共类

   a. 数据库配置文件，在resources目录下创建db.properties文件用于存储数据库配置信息

Properties

```
diver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306?useUnicode=true&characterEncoding=utf-8
username=root
password=123456
```

   b. 编写数据库的公共类

Java

```java
package com.kuang.dao;

import java.io.IOException;
import java.io.InputStream;
import java.sql.*;
import java.util.Properties;

//操作数据库的公共类
public class BaseDao {
    private static String driver;
    private static String url;
    private static String username;
    private static String password;

    //静态代码块，类加载的时候就初始化了
    static {
        Properties properties = new Properties();
        //通过类加载器读取对应的资源
        InputStream is = BaseDao.class.getClassLoader().getResourceAsStream("db.properties");

        try {
            properties.load(is);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        driver = properties.getProperty("driver");
        url = properties.getProperty("url");
        username = properties.getProperty("username");
        password = properties.getProperty("password");
    }

    //获取数据库链接
    public static Connection getConnection(){
        Connection connection = null;
        try {
            Class.forName(driver);
            connection = DriverManager.getConnection(url, username, password);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return connection;
    }
```

```
44
45
46      //编写查询公共类
        public static ResultSet execute(Connection connection,String sql,Object[] params,ResultSet resultSet,PreparedStatement preparedStatement) throws SQLException {
47          //prepareStatement 方法通过使用提供的 SQL 查询字符串来预编译 SQL 语句
48          //预编译的sql，在后面直接执行就可以了
49          preparedStatement = connection.prepareStatement(sql);
50
51          for (int i = 0; i < params.length; i++) {
52              //使用 setObject 方法将每个参数设置到 PreparedStatement 对象中的相应位置
53              //setObject,占位符从1开始，但是我们的数组是从0开始的!
54              preparedStatement.setObject(i+1,params[i]);
55          }
56          //这行代码使用 PreparedStatement 执行查询，返回一个 ResultSet 对象
57          resultSet = preparedStatement.executeQuery();
58          return resultSet;
59      }
60
61      //编写增删改公共方法
62      public static int execute(Connection connection,String sql,Object[] params,PreparedStatement preparedStatement) throws SQLException {
63          preparedStatement = connection.prepareStatement(sql);
64
65          for (int i = 0; i < params.length; i++) {
66              //setObject,占位符从1开始，但是我们的数组是从0开始的!
67              preparedStatement.setObject(i+1,params[i]);
68          }
69          //这行代码调用 executeUpdate 方法执行 SQL 更新操作，并将返回的受影响行数存储在 updateRows 变量中
70          int updateRows= preparedStatement.executeUpdate();
71          return updateRows;
72      }
73
74      //释放资源
75      public static void close(Connection connection, PreparedStatement preparedStatement, ResultSet resultSet){
76          boolean flag = true;
77          if (resultSet!= null){
78              try {
79                  resultSet.close();
80                  //GC回收
81                  resultSet = null;
82              } catch (SQLException e) {
83                  e.printStackTrace();
84                  flag = false;
85              }
```

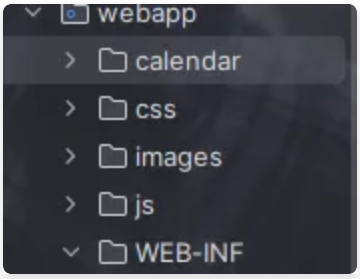
```
        }

        if (preparedStatement!= null){
            try {
                preparedStatement.close();
                //GC回收
                preparedStatement = null;
            } catch (SQLException e) {
                e.printStackTrace();
                flag = false;
            }
        }

        if (connection!= null){
            try {
                connection.close();
                //GC回收
                connection = null;
            } catch (SQLException e) {
                e.printStackTrace();
                flag = false;
            }
        }
    }
}
```

c. 编写字符编码过滤器

```java
package com.kuang.filter;

import javax.servlet.*;
import java.io.IOException;

public class CharacterEncodingFilter implements Filter {


    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    //这是过滤器的核心方法，每次请求都要经过这个方法进行处理。
    //参数 servletRequest 是 ServletRequest 对象，代表客户端请求。
    //参数 servletResponse 是 ServletResponse 对象，代表服务器对客户端的响应。
    //参数 filterChain 是 FilterChain 对象，它的 doFilter 方法将请求和响应传递到下一个过滤器或目标资源（如 Servlet 或 JSP）。
    //通常在这里编写拦截逻辑，然后使用 filterChain.doFilter(request, response) 来继续处理请求或响应。
    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterchain) throws IOException, ServletException {
        servletRequest.setCharacterEncoding("UTF-8");
        servletResponse.setCharacterEncoding("UTF-8");

        filterchain.doFilter(servletRequest, servletResponse);
    }

    @Override
    public void destroy() {

    }
}



<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0"
         metadata-complete="true">
```

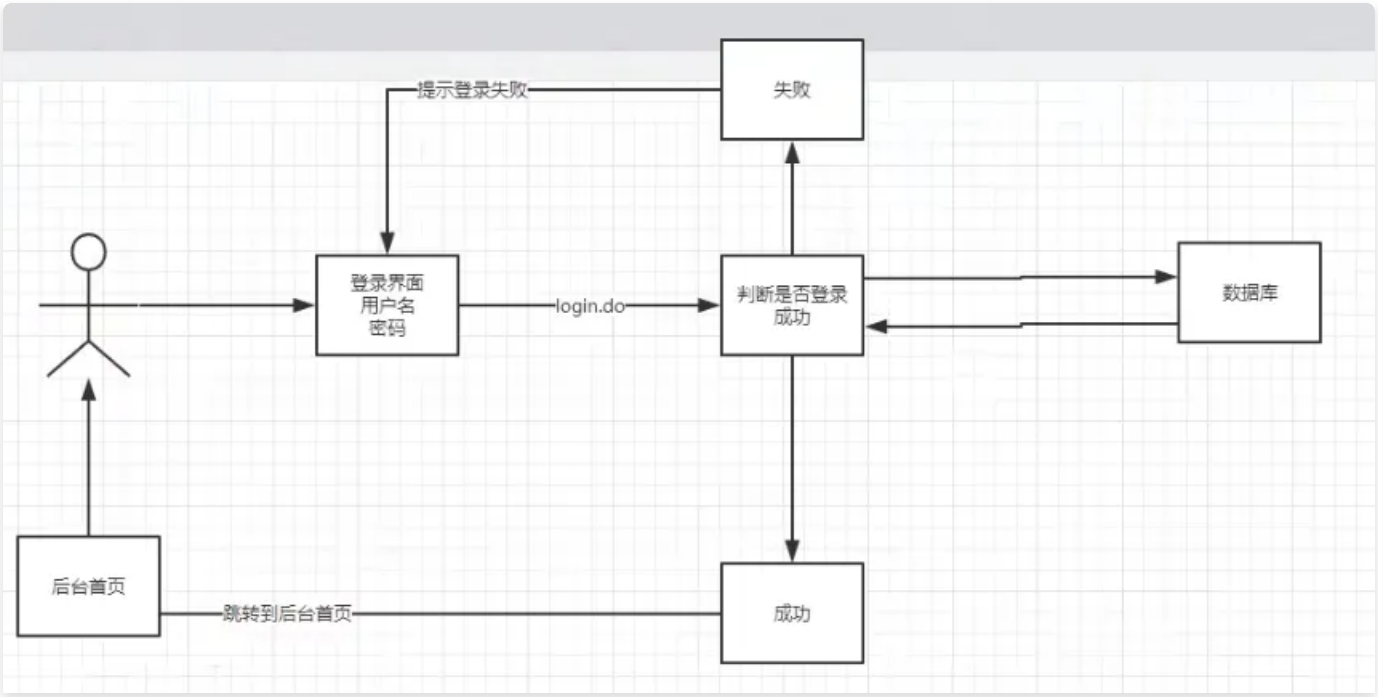
```
42
43      <!--字符编码-->
44      <filter>
45          <filter-name>CharacterEncodingFilter</filter-name>
46          <filter-class>com.kuang.filter.CharacterEncodingFilter</filter-cla
    ss>
47      </filter>
48      <filter-mapping>
49          <filter-name>CharacterEncodingFilter</filter-name>
50          <url-pattern>/*</url-pattern>
51      </filter-mapping>
52
53  </web-app>
```

8. 导入静态资源



登录功能实现

原理图



1. 编写前端页面

2. 设置首页

```xml
<!--设置欢迎页面-->
<welcome-file-list>
  <welcome-file>login.jsp</welcome-file>
</welcome-file-list>

```

3. 编写dao层登录用户登录的接口

```java
public interface UserDao {
    //得到要登录的用户
    //Connection 对象，表示数据库连接
    //第二个参数是 String，表示用户编码。这个编码通常是用户的唯一标识符，用于在数据库中查询特定用户的信息
    public User getLoginUser(Connection connection,String userCode) throws SQLException;
}
```

4. 编写dao接口的实习类

Java

```java
package com.kuang.dao.user;
import com.kuang.dao.BaseDao;
import com.kuang.pojo.User;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
//implements UserDao表示实现某个接口
public class UserDaoImpl implements UserDao {
    //得到要登录的用户
    @Override
    public User getLoginUser(Connection connection, String userCode) throws SQLException {

        PreparedStatement pstm = null;
        ResultSet rs = null;
        User user = null;

        if(connection!=null){
            //通过 "SELECT * from smbms_user where userCode=?"查询语句，查出sql
            String sql = "SELECT * from smbms_user where userCode=?";
            Object[] params = {userCode};


                rs = BaseDao.execute(connection, pstm, rs, sql, params);
                if(rs.next()) {
                    // 用户对象的初始化代码
                    //封装到user中
                    user =new User();
                    user.setId(rs.getInt("id"));
                    user.setUserCode(rs.getString("userCode"));
                    user.setUserName(rs.getString("userName"));
                    user.setUserPassword(rs.getString("userPassword"));
                    user.setGender(rs.getInt("gender"));
                    user.setBirthday(rs.getDate("birthday"));
                    user.setPhone(rs.getString("phone"));
                    user.setAddress(rs.getString("address"));
                    user.setUserRole(rs.getInt("userRole"));
                    user.setCreatedBy(rs.getInt("createdBy"));
                    user.setCreationDate(rs.getTimestamp("creationDate"));
                    user.setModifyBy(rs.getInt("modifyBy"));
                    user.setModifyDate(rs.getTimestamp("modifyDate"));
                }
                BaseDao.closeResource(null,pstm,rs);
```

```
        }
        return user;
    }
}
```

5. 业务层接口

```java
public interface UserService {
    //用户登录
    public User login(String userCode, String password);
}
```

6. 业务层实现类

```java
public class UserServiceImpl implements UserService {
    //业务层都会调用dao层，所以我们要引入Dao层；
    private UserDao userDao;
    public UserServiceImpl() {
        userDao = new UserDaoImpl();
    }
    @Override
    public User login(String userCode, String password) {
        Connection connection = null;
        User user = null;

        try {
            connection = BaseDao.getConnection();
            //通过业务层调用对应的具体的数据库操作
            user = userDao.getLoginUser(connection, userCode);
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            BaseDao.closeResource(connection,null,null);
        }
        if (null != user) {
            if (!user.getUserPassword().equals(password)) {
                user = null;
            }
        }
        return user;
    }
}
```

7. 编写Servlet

Java

```java
package com.kuang.servlet.user;

import com.kuang.pojo.User;
import com.kuang.service.user.UserService;
import com.kuang.service.user.UserServiceImpl;
import com.kuang.util.Constants;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("LoginServlet--start....");

        // 获取用户名和密码
        String userCode = req.getParameter("userCode");
        String userPassword = req.getParameter("userPassword");

        // 和数据库中的密码进行对比，调用业务层代码
        UserService userService = new UserServiceImpl();
        User user = userService.login(userCode, userPassword);

        if (user != null) { // 查有此人，可以登录
            req.getSession().setAttribute(Constants.USER_SESSION, user);
            resp.sendRedirect("jsp/frame.jsp");
            return; // 防止后续代码执行
        } else { // 查无此人，无法登录
            req.setAttribute("error", "用户名或者密码错误");
            req.getRequestDispatcher("login.jsp").forward(req, resp);
            return; // 防止后续代码执行
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);
    }
}

```

8. 注册Servlet

```xml
<servlet>
  <servlet-name>LoginServlet</servlet-name>
  <servlet-class>com.kuang.servlet.user.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>LoginServlet</servlet-name>
  <url-pattern>/login.do</url-pattern>
</servlet-mapping>
```

9. 测试访问，确保以上功能成功！

登录功能优化

注销功能：

思路：移除Session，返回登录页面

```java
public class LogoutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        //移除用户的Session
        req.getSession().removeAttribute(Constants.USER_SESSION);
        //返回登陆页面
        resp.sendRedirect("/login.jsp");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);
    }
}

```

注册xml

```xml
<servlet>
  <servlet-name>LogoutServlet</servlet-name>
  <servlet-class>com.kuang.servlet.user.LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>LogoutServlet</servlet-name>
  <url-pattern>/jsp/logout.do</url-pattern>
</servlet-mapping>
```

登录拦截优化

编写一个过滤器并注册

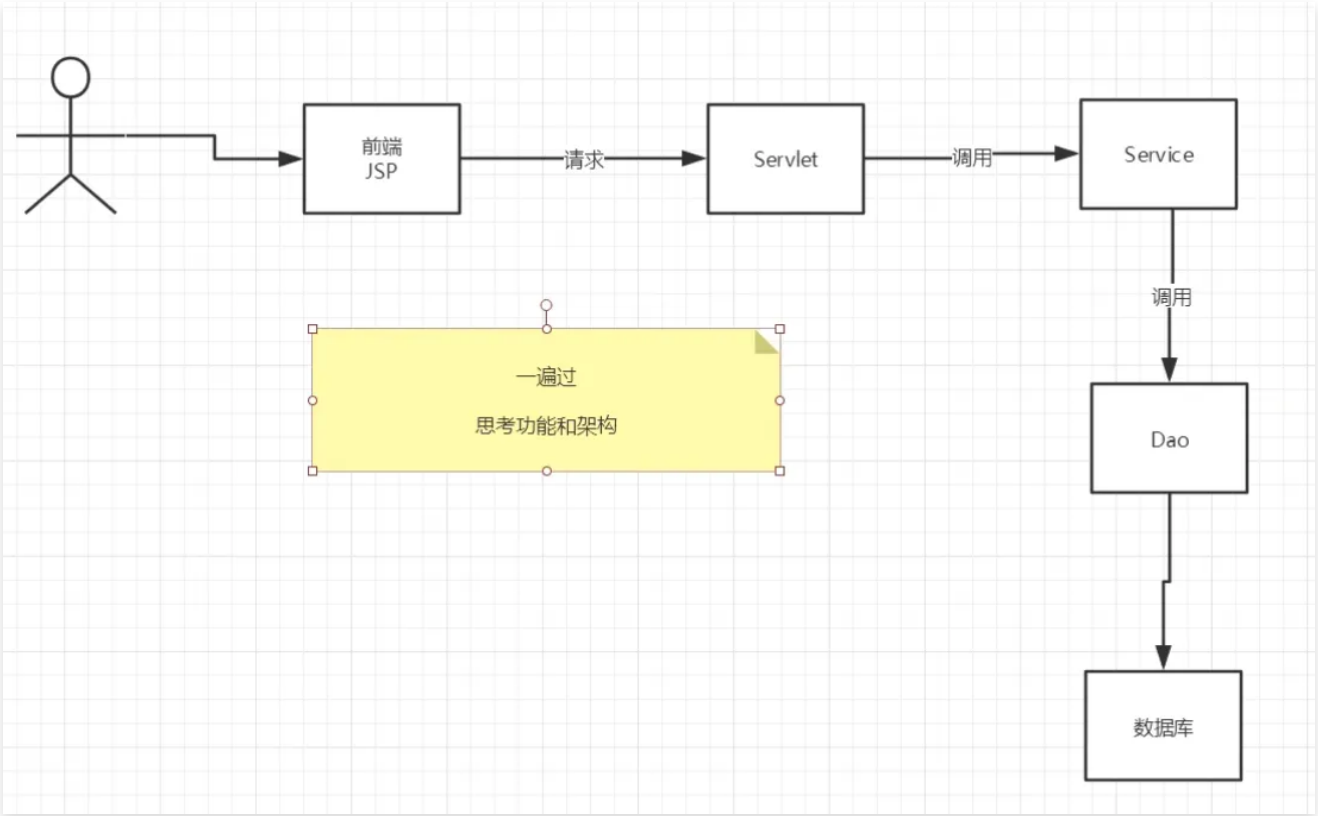
```java
public class SysFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) resp;

        //过滤器，从Session中获取用户
        User user = (User) request.getSession().getAttribute(Constants.USER_SESSION);

        if(user==null){//已经被移除或者注销了，或者未登录
            response.sendRedirect("/smbms/error.jsp");
        }else {
            chain.doFilter(req, resp); // 继续执行下一个过滤器或Servlet
        }
    }
    @Override
    public void destroy() {

    }
}
```

```xml
<!--用户登录过滤器-->
    <filter>
        <filter-name>SysFilter</filter-name>
        <filter-class>com.kuang.filter.SysFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>SysFilter</filter-name>
        <url-pattern>/jsp/*</url-pattern>
    </filter-mapping>
```

密码修改

1. 导入前端素材

```html
<li><a href="${pageContext.request.contextPath }/jsp/pwdmodify.jsp">密码修改</a></li>
```

2. 

3. UserDao接口

```java
//修改当前用户密码
public int updatePwd(Connection connection,int id,int password) throws SQLException;
```

4. UserDao接口实现类

```java

    //修改当前用户密码
    @Override
    public int updatePwd(Connection connection, int id, int password) throws SQLException {

        PreparedStatement pstm = null;
        int execute = 0;
        if (connection != null) {
            String sql = "update smbms_user set userPassword = ? where id = ?" ;
            Object params[] = {password, id};
            execute = BaseDao.execute(connection, sql, params, pstm);
            BaseDao.closeResource(null,pstm,null);
        }
        return execute;
    }
```

5. UserService层

```java
    //根据用户ID修改密码
    public boolean updatePwd(int id, int pwd);
```

6. UserService实现类

```java
@Override
    public boolean updatePwd(int id, int pwd) {
        //获取链接
        Connection connection = null;
        boolean flag = false;
        //修改密码
        try {
            connection = BaseDao.getConnection();
            if(userDao.updatePwd(connection,id,pwd)>0){
                flag = true;
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }finally {
            BaseDao.closeResource(connection,null,null);
        }
        return flag;
    }
```

7. Servlet记得实现复用，需要提出方法！

```java
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String method = req.getParameter("method");
        if(method.equals("savepwd")&&method!=null){
            this.updatePwd(req, resp);
        }
    }


public void updatePwd(HttpServletRequest req, HttpServletResponse resp){
        //从Session里面拿ID
        Object o = req.getSession().getAttribute(Constants.USER_SESSION);
        String newpassword = req.getParameter("newpassword");
        boolean flag = false;
        if(o!=null && newpassword!=null){
            UserServiceImpl userService = new UserServiceImpl();
            flag = userService.updatePwd(((User) o).getId(), newpassword);
            if(flag){
                req.setAttribute("message","修改密码成功，请退出，使用新密码登录");
                //密码修改成功，移除当前Session
                req.getSession().removeAttribute(Constants.USER_SESSION);
            }else{
                req.setAttribute("message","密码修改失败");
            }
        }else{
            req.setAttribute("message","新密码有问题");
        }

        try {
            req.getRequestDispatcher("pwdmodify.jsp").forward(req,resp);
        } catch (ServletException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

8. 测试

优化密码修改使用Ajax

1. 阿里巴巴fastjson

```xml
<!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.83</version>
</dependency>
```

2. 后台代码修改

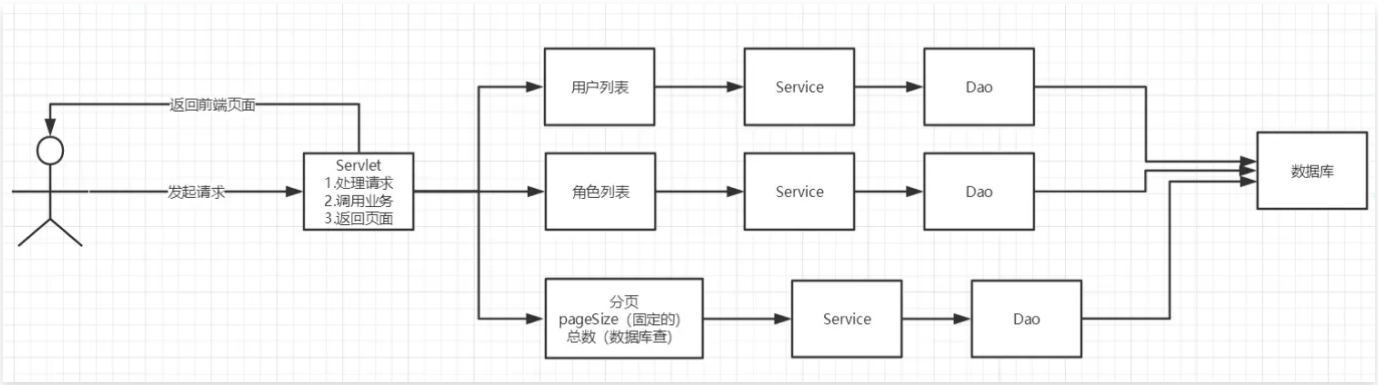
```java
//修改密码
public void updatePwd(HttpServletRequest req, HttpServletResponse resp){
    //从Session里面拿ID
    Object o = req.getSession().getAttribute(Constants.USER_SESSION);
    String newpassword = req.getParameter("newpassword");
    boolean flag = false;
    if(o!=null && newpassword!=null){
        UserServiceImpl userService = new UserServiceImpl();
        flag = userService.updatePwd(((User) o).getId(), newpassword);
        if(flag){
            req.setAttribute("message","修改密码成功，请退出，使用新密码登录");
            //密码修改成功，移除当前Session
            req.getSession().removeAttribute(Constants.USER_SESSION);
        }else{
            req.setAttribute("message","密码修改失败");
        }
    }else{
        req.setAttribute("message","新密码有问题");
    }

    try {
        req.getRequestDispatcher("pwdmodify.jsp").forward(req,resp);
    } catch (ServletException e) {
        throw new RuntimeException(e);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

//验证旧密码,session中有用户的密码
public void pwdModify(HttpServletRequest req, HttpServletResponse resp){
    //从Session里面拿ID
    Object o = req.getSession().getAttribute(Constants.USER_SESSION);
    String oldpassword = req.getParameter("oldpassword");

    //万能的Map：结果集
    Map<String,String> resultMap = new HashMap<String,String>();

    if(o==null){//session失效了，session过期了
        resultMap.put("result","sessionerror");
    } else if (StringUtils.isNullOrEmpty(oldpassword)) {//输入的密码为空
        resultMap.put("result","error");
    } else{
        String userPassword = ((User) o).getUserPassword();//Session中用户的密码
```

```
            if(oldpassword.equals(userPassword)){
                resultMap.put("result","true");
            }else {
                resultMap.put("result","true");
            }
        }

        try {
            resp.setContentType("application/json");
            PrintWriter writer = resp.getWriter();
            //JSONArray 阿里巴巴的JSON工具类，转换格式
            writer.write(JSONArray.toJSONString(resultMap));
            writer.flush();
            writer.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }
```

3. 测试

用户管理实现

思路：



1. 导入分页的工具类
2. 用户列表页面导入 userlist.jsp

1.获取用户数量

1. UserDao

```java
//根据用户名或者角色查询用户总数
    public int getUserCount(Connection connection,String userName,int userR
ole )throws SQLException;
```

2. UserDaoImpl

```java
    //根据用户名或者角色查询用户总数
    @Override
    public int getUserCount(Connection connection, String username, int userRole) throws SQLException {
        PreparedStatement pstm = null;
        ResultSet rs = null;
        int count = 0;
        if(connection != null) {
           StringBuffer sql = new StringBuffer();
           //append表示追加
           sql.append("select count(1) as count from smbms_user u,smbms_role r where u.userRole = r.id");
            ArrayList<Object> list = new ArrayList<>();//存放我们的参数

            if(!StringUtils.isNullOrEmpty(username)){
                sql.append(" and u.userName like ?");
                list.add("%"+username+"%");//index:0
            }
            if(userRole > 0){
                sql.append(" and u.userRole = ?");
                list.add(userRole);//index:1
            }

            //怎么把List转换为数组
            Object[] params = list.toArray();
            System.out.println("UserDaoImpl->getUserCount"+sql.toString());//输出最后完整的sql语句
            rs = BaseDao.execute(connection, pstm, rs, sql.toString(), params);

            if(rs.next()){
                count = rs.getInt("count");//从结果集中获取最终的数量
            }

            BaseDao.closeResource(null, pstm, rs);
        }

        return count;
    }
```

3. UserService

```java
    //查询记录数
    public int getUserCount(String userName,int userRole);
```

4. UserServiceImpl

```java
    //查询记录数
    @Override
    public int getUserCount(String userName, int userRole) {
        Connection connection = null;
        int count = 0;
        try {
            connection = BaseDao.getConnection();
            count = userDao.getUserCount(connection,userName,userRole);
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            BaseDao.closeResource(connection,null,null);
        }
        return count;
    }



    @Test
    public void test(){
        UserServiceImpl userService = new UserServiceImpl();
        int userCount = userService.getUserCount(null, 2);
        System.out.println(userCount);
    }
```

2.获取用户列表

1. UserDao

```java
//通过条件查询-userList
    public List<User> getUserList(Connection connection,String userName,int userRole,int currentPageNo,int pageSize)throws SQLException;
```

2. UserDaoImpl

Java

```java
    @Override
    public List<User> getUserList(Connection connection, String userName, int userRole, int currentPageNo, int pageSize) throws SQLException {
        PreparedStatement pstm = null;
        ResultSet rs = null;
        List<User> userList = new ArrayList<User>();

        if (connection != null) {
            StringBuffer sql = new StringBuffer();
            sql.append("select u.*,r.roleName as userRoleName from smbms_user u,smbms_role r where u.userRole = r.id");
            List<Object> list = new ArrayList<Object>();

            if (!StringUtils.isNullOrEmpty(userName)) {
                sql.append(" and u.userName like ?");
                list.add("%" + userName + "%");
            }
            if (userRole > 0) {
                sql.append(" and u.userRole = ?");
                list.add(userRole);
            }

            //在mysql数据库中，分页使用 limit startIndex，pageSize ； 总数
            //当前页     （当前页-1）*页面大小
            //0,5           1  0   01234
            //5,5           2  5   56789
            //10,5          3  10  10-14
            sql.append(" order by creationDate DESC limit ?,?");
            currentPageNo = (currentPageNo - 1) * pageSize;
            list.add(currentPageNo);
            list.add(pageSize);

            Object[] params = list.toArray();
//            System.out.println("sql ----> " + sql.toString());
            rs = BaseDao.execute(connection, pstm, rs, sql.toString(), params);
            while (rs.next()) {
                User _user = new User();
                _user.setId(rs.getInt("id"));
                _user.setUserCode(rs.getString("userCode"));
                _user.setUserName(rs.getString("userName"));
                _user.setGender(rs.getInt("gender"));
                _user.setBirthday(rs.getDate("birthday"));
                _user.setPhone(rs.getString("phone"));
                _user.setUserRole(rs.getInt("userRole"));
```

```
                _user.setUserRoleName(rs.getString("userRoleName"));
                userList.add(_user);
            }
            BaseDao.closeResource(null, pstm, rs);
        }
        return userList;
    }
```

3. UserService

```Java
//根据条件查询用户列表
    public List<User> getUserList(String queryUserName, int queryUserRole,
int currentPageNo, int pageSize);
```

4. UserServiceImpl

```Java
@Override
    public List<User> getUserList(String queryUserName, int queryUserRole,
 int currentPageNo, int pageSize) {
        Connection connection = null;
        List<User> userList = null;
//        System.out.println("queryUserName ----> " + queryUserName);
//        System.out.println("queryUserName ----> " + queryUserRole);
//        System.out.println("currentPageNo ----> " + currentPageNo);
//        System.out.println("pageSize ----> " + pageSize);
        try {
            connection = BaseDao.getConnection();
            userList = userDao.getUserList(connection, queryUserName, quer
yUserRole, currentPageNo, pageSize);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            BaseDao.closeResource(connection, null, null);
        }
        return userList;
    }
```

3.获取角色操作!

为了我们职责统一，可以把角色的操作单独放在一个包中，和pojo类对应==

RoleDao

```java
public interface RoleDao {

    //获取角色列表
    public List<Role> getRoleList(Connection connection)throws SQLException;
}
```

RoleDaoImpl

```java
public class RoleDaoImpl implements RoleDao {
    //获取角色列表
    @Override
    public List<Role> getRoleList(Connection connection) throws SQLException {
        PreparedStatement pstm =null;
        ResultSet resultSet = null;
        ArrayList<Role> roleList = new ArrayList<>();
        if(connection != null){
            String sql = "select * from smbms_role";
            Object[] params = {};
            resultSet = BaseDao.execute(connection, pstm, resultSet, sql, params);

            while (resultSet.next()){
                Role _role = new Role();
                _role.setId(resultSet.getInt("id"));
                _role.setRoleCode(resultSet.getString("roleCode"));
                _role.setRoleName(resultSet.getString("roleName"));
                roleList.add(_role);
            }
            BaseDao.closeResource(null, pstm, resultSet);
        }
        return roleList;
    }
}

```

RoleService

```java
public interface RoleService {
    //获取角色列表
    public List<Role> getRoleList();
}
```

RoleService

```java
public class RoleServiceImpl implements RoleService{
    //引入Dao
    private RoleDao roleDao;
    public RoleServiceImpl() {
        roleDao = new RoleDaoImpl();
    }

    @Override
    public List<Role> getRoleList() {
        Connection connection= null;
        List<Role> roleList = null;
        try {
            connection = BaseDao.getConnection();
            roleList = roleDao.getRoleList(connection);
        }catch (Exception e) {
            e.printStackTrace();
        }finally {
            BaseDao.closeResource(connection, null, null);
        }
        return roleList;
    }
}

```

4.用户显示的Servlet

1. 获取用户前端的数据 （查询）
2. 判断请求是否需要执行，看参数的值判断
3. 为了实现分页，需要计算出当前页面和总页面，页面大小
4. 用户列表展示
5. 返回前端

Java

```java
//查询 重点难点
    public void query(HttpServletRequest req, HttpServletResponse resp) {

        // 查询用户列表

        // 从前端获取数据:
        String queryUserName = req.getParameter("queryname");
        String tmp = req.getParameter("queryUserRole");
        String pageIndex = req.getParameter("pageIndex");
        int queryUserRole = 0;

        // 获取用户列表
        UserServiceImpl userService = new UserServiceImpl();
        List<User> userList = null;

        // 获取配置的页面大小，默认为5
        int pageSize = 5; // 默认页面大小
        String pageSizeStr = getServletContext().getInitParameter("pageSiz
e");
        if (pageSizeStr != null && !pageSizeStr.isEmpty()) {
            pageSize = Integer.parseInt(pageSizeStr);
        }

        int currentPageNo = 1; // 默认当前页码

        // 如果查询用户名为空，则设置为空字符串
        if (queryUserName == null) {
            queryUserName = "";
        }

        // 如果角色ID不为空，解析为整数
        if (tmp != null && !tmp.isEmpty()) {
            try {
                queryUserRole = Integer.parseInt(tmp);
            } catch (NumberFormatException e) {
                // 处理异常，例如记录日志或给出默认值
                queryUserRole = 0; // 或者您可以选择返回错误信息
            }
        }

        // 如果页面索引不为空，解析为整数
        if (pageIndex != null && !pageIndex.isEmpty()) {
            try {
                currentPageNo = Integer.parseInt(pageIndex);
            } catch (NumberFormatException e) {
```

```
                // 处理异常，例如记录日志或给出默认值
                currentPageNo = 1; // 或者您可以选择返回错误信息
            }
        }

        // 获取用户总数
        int totalCount = userService.getUserCount(queryUserName, queryUserRole);

        // 总页数支持
        PageSupport pageSupport = new PageSupport();
        pageSupport.setCurrentPageNo(currentPageNo);
        pageSupport.setPageSize(pageSize);
        pageSupport.setTotalPageCount((int) Math.ceil(totalCount * 1.0 / pageSize));

        int totalPageCount = pageSupport.getTotalPageCount();

        // 控制首页和尾页
        if (currentPageNo < 1) {
            currentPageNo = 1;
        } else if (currentPageNo > totalPageCount) {
            currentPageNo = totalPageCount;
        }

        // 获取用户列表展示
        userList = userService.getUserList(queryUserName, queryUserRole, currentPageNo, pageSize);
        req.setAttribute("userList", userList);

        // 获取角色列表
        RoleServiceImpl roleService = new RoleServiceImpl();
        List<Role> roleList = roleService.getRoleList();
        req.setAttribute("roleList", roleList);
        req.setAttribute("totalCount", totalCount);
        req.setAttribute("currentPageNo", currentPageNo);
        req.setAttribute("totalPageCount", totalPageCount);
        req.setAttribute("queryUserName", queryUserName);
        req.setAttribute("queryUserRole", queryUserRole);

        // 返回前端
        try {
            req.getRequestDispatcher("userlist.jsp").forward(req, resp);
        } catch (ServletException | IOException e) {
            e.printStackTrace();
        }
    }
```

一切的增删改操作都需要处理事务!

ACID